IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Number	09/883,508	Confirmation No.:	8696
Applicant	Jeffrey A. Bedell *et al.*		
Filed	June 19, 2001		
Title	System and Method for Managing Objects Between Projects		
TC/Art Unit	2194		
Examiner:	Li B. Zhen		
Docket No.	53470.003042		
Customer No.	**21967**		

APPEAL BRIEF

TABLE OF CONTENTS

Page

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Number	09/883,508	Confirmation No.:	8696
Applicant	Jeffrey A. Bedell *et al.*		
Filed	June 19, 2001		
Title	System and Method for Managing Objects Between Projects		
TC/Art Unit	2194		
Examiner:	Li B. Zhen		
Docket No.	53470.003042		
Customer No.	**21967**		

APPEAL BRIEF

In response to the Office Action dated July 26, 2006 finally rejecting pending claims 1-18, Appellant respectfully requests that the Board of Patent Appeals and Interferences reconsider and reverse the rejections of record, which are attached hereto as an Appendix.

I. Real Party In Interest

The real party in interest is Microstrategy, Incorporated as assignee of the entire interest in the above-referenced application, assigned by its inventors.

II. Related Appeals And Interferences

There are no known related appeals.

III. Status Of Claims

Claims 1, 3-5, 10-12 and 16-18 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over U.S. Patent No. 5,854,932 to Mariani *et al.* ("Mariani") in view of U.S. Patent No. 6,167,563 to Fontana *et al.* ("Fontana"). Claims 2, 6-9 and 13-15 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Mariani and Fontana in further view of U.S. Patent No. 6,112,024 to Almond *et al.* ("Almond"). The rejection of claims 1-18 is appealed.

IV. Status Of Amendments

No amendments to the claims have been filed subsequent to the final rejection dated July 26, 2006.

V. Summary Of Claimed Subject Matter

Appellant believes that a brief discussion of the embodiments of the invention and the problems solved by the embodiments of the present invention, will assist the Board of Patent Appeals and Interferences (hereinafter referred to as "the Board") in appreciating the significant advances made by the embodiments of the present invention.

A. Explanation of Independent Claim 1

A computer implemented method for managing groups of objects for use in a reporting system project comprising the steps of: (*See e.g.*, Figures 1-3, and 5; Page 2, lines 2-3; Page 3, lines 11-21)

receiving a command to perform a selected function on a selected object; (Figure 2, reference character 204; Page 4, lines 4-5; Page 14, lines 8-10; Page 16, lines 12-17)

automatically identifying dependent objects referred to by the selected object; (Figure 3, reference character 304; Page 15, lines 1-14; Page 20, lines 2-5)

determining using a computer processor an appropriate manner of executing the selected function on the selected object; (Figures 3 -5; Page 16, lines 17-21; Page 17, lines 10-17; Page 18, line 15 - Page 19, line 5)

determining using a computer processor appropriate functions to be performed on the dependent objects; (Figures 3 -5; Page 17, lines 14-21; Page 19, line 17 - Page 20, line 6)

automatically causing the appropriate functions to be performed on the dependent objects; and (Figures 3 -5; Page 17, lines 14-21; Page 19, line 17 - Page 20, line 6)

automatically causing the execution of the selected function on the selected object in the appropriate manner. (Figures 3 -5; Page 17, lines 14-21; Page 19, line 17 - Page 20, line 6)

B. Explanation of Independent Claim 10

A system application for managing objects within and between projects of a reporting system, the objects including unique identifiers and version identifiers that are similar between projects, the system application comprising: (*See e.g.*, Figures 2-5; Page 2, lines 2-7; Page 3, lines 11-21)

a user interface for receiving a user command to manipulate a selected object; and (*See e.g.*, Figures 1-2; Page 6, lines 14-17; Page 11, lines 14-15)

an operational module interfacing with the projects for identifying dependent objects referred to by the selected object, determining an appropriate manner of executing the user command, determining appropriate functions to be performed on the dependent objects, performing the appropriate functions on dependent objects, and executing the user command in the appropriate manner. (*See e.g.*, Figures 3-5; Page 14, line 18 - Page 15, line 5; Page 17, lines 18-21; Page 18, line 15 - Page 19, line 16)

C. Explanation of Independent Claim 18

A processor-readable medium including code for execution on a processor to managing groups of objects comprising: (*See e.g.*, Figures 3-5; Page 2, lines 1-7; Page 3, lines 11-21)

code for receiving a command to perform a selected function on a selected object; (*See e.g.*, Figures 3-5; Page 15, line 15 - Page 16, line 2; Page 17, lines 8-21; Page 18, line 15 - Page 19, line 16)

code for identifying dependent objects referred to by the selected object; (Figure 3, reference character 304; Page 15, lines 1-14; Page 20, lines 2-5)

code for determining an appropriate manner of executing the selected function; (Figures 3 -5; Page 17, lines 14-21; Page 19, line 17 - Page 20, line 6)

code for determining appropriate functions to be performed on the dependent objects; (Figures 3 -5; Page 17, lines 14-21; Page 19, line 17 - Page 20, line 6)

code for causing the appropriate functions to be performed on the dependent objects; and (Figures 3 -5; Page 17, lines 14-21; Page 19, line 17 - Page 20, line 6)

code for causing the execution of the selected function in the appropriate manner. (Figures 3 -5; Page 17, lines 14-21; Page 19, line 17 - Page 20, line 6)

VI. Grounds Of Rejection To Be Reviewed On Appeal

The following grounds of rejection are to be reviewed on appeal:

1) The rejection under 35 U.S.C. § 103(a) of claims 1, 3-5, 10-12 and 16-18 based on Mariani in view of Fontana.

2.) The rejection under 35 U.S.C. § 103(a) of claims 2, 6-9 and 13-15 based on Mariani and Fontana in further view of Almond.

VII. Argument

A. The Rejection Under 35 U.S.C. § 103(a) of Claims 1, 3-5, 10-12 and 16-18 based on Mariani in View of Fontana is Improper

Claims 1, 3-5, 10-12 and 16-18 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Mariani in view of Fontana. This rejection is improper.

Under 35 U.S.C. § 103, the Patent Office bears the burden of establishing a prima facie case of obviousness. In re Fine, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). The Patent Office can satisfy this burden only by showing some objective teaching in the prior art or that knowledge generally available to one of ordinary skill in the art would lead that individual to combine the relevant teachings of references. Id.. Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. ACS Hospital Systems, Inc. v. Montefiore Hospital, 732 F.2d 1572, 1577, 221 USPQ 929, 933 (Fed. Cir. 1984). That is, under 35 U.S.C. § 103, teachings of references can be combined only if there is some suggestion or motivation to do so. Id.. However, the motivation cannot come from the applicant's invention itself. In re Oetiker, 977 F.2d 1443, 1447, 24 USPQ2d 1443, 1446 (Fed. Cir. 1992). Rather, there must be some reason, suggestion, or motivation found in the prior art whereby a person of ordinary skill in the art would make the combination. Id..

As stated in MPEP § 2143, to establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in applicant's disclosure. In re Vaeck, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991). Also, as stated in MPEP § 2143.01, obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or

motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988); In re Jones, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. In re Mills, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). Further, as stated in MPEP § 2143.03, to establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. In re Royka, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). That is, "[a]ll words in a claim must be considered in judging the patentability of that claim against the prior art." In re Wilson, 424 F.2d 1382, 165 USPQ 494, 496 (CCPA 1970).

The Office Action of February 3, 2006 alleges that Mariani teaches the invention substantially as claimed with the exception of "automatically identifying dependent objects, automatically causing appropriate functions to be performed on the dependent objects and automatically causing execution of the selected function on the selected object." (Office Action, Page 4) For this element, the Office Action relies on Fontana. The Final Office Action maintains this argument. This rejection is improper for several reasons.

1. Mariani Fails to Disclose Many of the Recitations of Claim 1.

First, Mariani does not teach or suggest "determining using a computer processor an appropriate **manner** of executing the selected function" as recited in limitation (iii) of claim 1. The Office Action of February 3, 2006 asserts that determining when a function (e.g., recompiling) can be avoided (or not avoided) amounts to determining an "appropriate manner" of performing the function. The word "manner" refers to the way that something is done, not the fact of whether or not it is done. Similarly, the limitation "determining whether to execute a

selected function" is not an example of "determining...an appropriate **manner** of executing the selected function," as recited in claim 1. To assert that a "manner" can be taught by a simple "yes or no" determination strips the word "manner" of any meaning. To put it another way, an "appropriate manner" of performing a function cannot be to perform the function, nor can it be to not perform the function.

The Final Office Action maintains this rejection on the same argument. In response to the Appellant's arguments, the Office continues to assert that Mariani "teaches determining using a computer processor an appropriate manner of executing the selected function." (Final Office Action, page 3). To support its allegation, the Office asserts that "determining how the object code files are dependent on header files and detecting changes to the header files [col. 9, lines 1 - 15 of Mariani]" discloses "determining using a computer processor an appropriate manner of executing the selected function." (Final Office Action, page 3). The portion of Mariani cited is important to see in context.

> Referring now to FIG. 3, a minimal rebuild system 100 according to the exemplary embodiments of the invention is integrated into the development environment 52 (FIG. 2). The minimal rebuild system 100 operates to reduce rebuilding of the executable program 64 (FIG. 2) by avoiding recompiling the source code files 60 (FIG. 2) and the header files 62 (FIG. 2) into the object code files 82 (FIG. 2) where any changes to the header files 62 do not affect the resulting object code files 82. As shown in FIG. 4, the minimal rebuild system 100 determines when recompiling can be avoided by determining how the object code files 82 are dependent on the header files 62 ("dependency analysis" 112), and detecting changes to the header files 62 ("change detection" 114). The minimal rebuild system 100 can then compare each object code file's dependencies with the detected changes ("comparison of dependencies and changes" 116) and avoid recompiling the object code file when the object code file's dependencies and the detected changes do not intersect (as shown at step 118).

(Mariani, Col. 8, line 64 - Col. 9, line 15).

The Final Office Action, refers to object files, header files, changed files, dependent files and the executable program and asserts that a comparison of dependencies and changes to header files discloses "determining using a computer processor an appropriate manner of executing the selected function on the selected object." Appellant notes that the actions proposed by the Office to allegedly disclose the claim limitation would be redundant with other limitations in the claims. "[I]dentifying dependent objects referred to by the selected object" is recited in the second limitation of claim 1. Thus "determining how the object code files are dependent on the header files" can not disclose "determining using a computer processor an appropriate manner of executing the selected function on the selected object," as recited in the third limitation of claim 1. Furthermore, "detecting changes to the header files" is shown in the context of the cited passage above to be used by the minimal rebuild system of Mariani to avoid recompiling "where any changes to the header files 62 do not affect the resulting object code files 82." (Mariani, col. 9, lines 3-6). Avoiding recompiling is at best a determination of whether or not to execute a selected function and not "an appropriate manner of executing the selected function on the selected object."

The cited features of Mariani also fail to teach or suggest the elements of claim 1 because they fail to maintain a consistent notion of the recited claim elements (in particular, the recited nouns such as "selected object"). In other words, the Office Action of February 3, 2006 did not show, and the Final Rejection also fails to show a single, coherent embodiment of Mariani for all of the elements for which it is cited. Rather, the Office Action applies disparate features of Mariani that accomplish dissociated tasks. For instance, as to the "selected function" recitation of claim 1, the Office Action alternately applies "modifying code" in element (i) and "recompiling" in elements (iii) and (vi). Similarly, as to a "selected object," the Office Action

appears to apply a "source code file" in element (i), an "object code file" in element (ii), and specific source code files that were changed since the last project build in element (vi).

In the Final Office Action, the Office responds to earlier arguments by asserting that "the recompiling function is part of the modification function." (Final Office Action, page 3). Appellant notes that the cited language may disclose that modifying code may trigger recompilation. However, the two different functions are not one function as required by the claims.

Similarly, in response to earlier arguments that the Office alternatively relies on source code file and object code file to allegedly disclose a "selected object", the Final Office Action asserts that there is a relationship between source code files, object code files and the header files. Appellant notes that a relationship between three different types of items does not disclose a "selected object" as recited in the claims.

2. Fontana Fails to Cure the Deficiencies of Mariana.

As discussed above, Mariani fails to disclose at least the limitations directed to "automatically identifying dependent objects", "automatically causing the appropriate functions to be performed on the dependent objects" and "automatically causing the execution of the selected function on the selected object in the appropriate manner." Fontana fails to teach the missing limitations of Mariani.

Fontana discloses "an inquiry is made as to whether or not the user wants to update dependent components .. if the answer to this inquiry is no, then the components and dependent components [are not updated]" (Fontana Column 7, lines 27-30) Fontana further discloses "[After response to the inquiry] if the user does want to update dependent components [then the components are updated]" (Fontana Column 7, Lines 35-53). Appellant respectfully submits that

updating dependent components in response to a user prompt does not satisfy the "automatically identifying dependent objects," "automatically causing the appropriate functions to be performed on the dependent objects" and "automatically causing the execution of the selected function on the selected object in the appropriate manner" recitations.

In response to earlier arguments, the Final Office Action maintains the argument that Fontana allegedly discloses "automatically causing appropriate functions to be performed on the dependent objects [if the user does not want to update dependent components (yes leg of the diamond 66), then the dependent components are retrieved from the source control program." (Final Office Action, page 4). As cited above, it is clear that Fontana at best discloses updating dependent components in response to a user prompt and thus does not disclose "automatically causing the appropriate functions to be performed on the dependent objects" as required by the claims. **Requiring user input to automatically update dependent components is not the same as "automatically causing appropriate functions to be performed on the dependent objects."**

3. The Office Fails to Provide a Motivation or Suggestion to Modify Mariani in View of Fontana.

The mere fact that Mariani can be modified does not render the resultant modification obvious unless there is a suggestion or motivation found somewhere in the prior art regarding the desirability of the combination or modification. *See* M.P.E.P § 2143.01*; see also In re Mills*, 16 U.S.P.Q.2d 1430, 1432 (Fed. Cir. 1990); *In re Fritz*, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). In addition, the teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in Applicants' disclosure. *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991).

In *In re Hedges*, 783, F.2d 1038, 1041, 228 U.S.P.Q. 685, 687, (Fed. Cir. 1986), the U.S. Court of Appeals for the Federal Circuit stated that "the prior art as a whole must be considered. The teachings are to be viewed as they would have been viewed by one of ordinary skill." The court also stated that "'[i]t is impermissible within the framework of section 103 to pick and choose from any one reference only so much of it as will support a given position, to the exclusion of other parts necessary to the full appreciation of what such reference fairly suggests to one of ordinary skill in the art'" (quoting *In re Wesslau*, 353 F.2d 238, 241, 147 U.S.P.Q. 391, 393 (CCPA, 1965)).

The Office asserts that "it would have been obvious to a person of ordinary skill in the art at the time of the invention to apply the teaching of automatically identifying dependent objects and automatically causing appropriate functions to be performed on the dependent objects and automatically causing execution of the selected function on the selected object" and cites col. 1, lines 33-37 of Fontana for alleged support. (Final Office Action, page 8). The Office further asserts that the motivation would be to provide "a method for building or modifying software components inside a computer system and updating all dependent components automatically in a manner transparent to the user and the computer system." (Final Office Action, page 8). The cited portion of Fontana allegedly discloses updating all dependent components automatically. Updating dependent components automatically **is not** "automatically causing the appropriate functions to be performed on the dependent objects." As discussed above, Mariani fails to disclose "automatically causing the appropriate functions to be performed on the dependent objects." Fontana's alleged disclosure of automatically updating all dependent components does not remedy this deficiency so the Office's proposed motivation fails. Fontana, as discussed

above, also does not remedy the other deficiencies of Mariani. Therefore, any motivation to combine Fontana with Mariani to remedy the deficiencies of Mariani would fail.

Accordingly, the Office has failed to provide any proper motivation for modifying Mariani, so the proposed modification fails.

Claim 18 contains similar limitations to claim 1 and is thus allowable for at least the above reasons.

B. Claim 10 is Separately Patentable

Claim 10 additionally recites "a system application for managing objects within and between projects of a reporting system, the objects including unique identifiers and version identifiers that are similar between projects." The Final Office Action rejects claim 10 as a system claim that corresponds to method claim 1. Appellants note that "managing objects with and **between projects of a reporting system**," has not been addressed. There is no disclosure of managing objects between projects of a reporting system in Mariani.

Thus, Mariani fails to teach claim 1 as well as claims 10 and 18, which have some related limitations. The same arguments apply to claims 3-5, 11, 12, 16, and 17, which depend from and incorporate the limitations of claims 1 and 10.

For at least the foregoing reasons, the §103(a) rejections of all of claims 1-18 cannot stand and should be overturned.

C. Claim 4 is Separately Patentable

Claim 4 recites "a selected function [which] relates to manipulating objects within and between projects and wherein within each project each object has a unique identifier and a version identifier." The Final Office Action alleges that this is disclosed by Mariani. A reference which does not even contain the term "version control" certainly does not disclose a method or system wherein " wherein within each project each object has a unique identifier and a

version identifier." The Office Action relies on a "name in the scope of a class" to allegedly disclose a unique identifier. (Final Office Action, page 9). However, even if a name in a scope of a class disclosed a unique identifier (which it does not), a class is not a project. Furthermore, Mariani does not disclose manipulating objects between projects.

For at least these additional reasons, the rejection of claim 4 should be overturned.

D. Claim 11 is Separately Patentable

Claim 11 recites an "operational module [which] interfaces with projects that reside in various environments." The Final Office Action alleges that claim 11 is disclosed by col. 8, lines 23-43 and col. 12, lines 28-67 of Mariani. The cited portion of Mariani at best discloses the presence of a development environment and a user's project. There is no disclosure of a system "wherein the operational module interfaces with projects that reside in various environments," as required by claim 11. The disclosure of a development environment is not the same as the disclosure of a system that interfaces with projects residing in various environments.

For at least these additional reasons the rejection of claim 11 should be overturned.

E. The Rejection of Claims 2, 6-9 and 13-15 Under 35 U.S.C. § 103(a) Based on Mariani and Fontana in Further View of Almond is Improper

Claims 2, 6-9 and 13-15 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Mariani and Fontana in further view of U.S. Patent No. 6,112,024 to Almond *et al.* ("Almond"). This rejection is also improper.

In regards to claim 2, the Office admits that "Mariani as modified does not teach the selected object is contained in metadata of an on-line analytical processing system." (Final Office Action, page 11.) The Office asserts that Almond remedies this admitted deficiency.

The Office Action of February 3, 2006 alleges that the meta model disclosed by Almond as "a schema - meta model - which facilitates version control. .. In essence, the model serves as a

container which facilitates version control." Almond, Column 3, lines 1-6 The Final Office Action maintains this argument. The present application indicates what is meant by "metadata" in this context:

> The access of data resources as described above relies on modules, reports, documents, prompts, filters, templates, metrics, custom groups, consolidations, searches, attributes, facts, hierarchies, transformations, partitions, tables, functions, users, database instances, schedules, etc. These data are distinct from the data are distinct from the data resources that are hosted and accessed by the system. These data that are created, stored and used by the system during its operation are referred to as metadata.

Page 12, lines 19-21 - Page 13, lines 1-3.

Appellant respectfully submits that the meta model disclosed by Almond does not teach "the selected object is contained in metadata of an on-line analytical processing system." Version control is not the same as metadata of an on-line analytical processing system. For at least these additional reasons, the rejection of claim 2 should be overturned.

F. Claim 6 is Separately Patentable

In regards to claim 6, the Office Action of February 3, 2006 asserts that Mariani as modified teaches the limitation of receiving a command to copy a selected object from a source project to a destination project. In particular, the Office Action cites a passage of Almond stating "the user will perform versioning activities such as...Get -- copy one or multiple objects to the user's local directory." Almond, col. 39, lines 28-32. The Final Action maintains this argument.

Almond's reference to copying objects fails to teach the subject matter of claim 6 because this feature cannot be combined with Mariani in the manner prescribed by claim 6.

Claim 6 recites "wherein the step of receiving is a step of receiving a command to copy a selected object from a source project to a destination project." Claim 6 also incorporates the limitations of claim 1, which in the context of claim 6 require the "identifying," "determining...manner," "determining...functions," "causing...functions," and "causing the execution" actions to be performed in accordance with the "command to copy a selected object from a source project to a destination project." Mariani teaches away from a combination with Almond in this context. The Office Action uses Mariani's system for selectively recompiling object code to purportedly establish how Mariani teaches the elements of claim 1. However, while the Mariani system is specifically designed to selectively recompile object code, it is not configured to copy objects in the manner by which it recompiles object code. Copying and recompiling are different operations that require different systems and methods. Therefore, even if Mariani taught the limitations of claim 1 (which it does not), the Mariani system cannot be combined with Almond to copy objects in the way that it selectively recompiles object code. Thus, the combination of Mariani and Almond does not teach or suggest claim 6.

G. Claim 7 is Separately Patentable

Claim 7 recites "the unique identifier and a version identifier of objects in a source project that are similar to the unique identifier and a version identifier of objects in the destination project." The Final Office Action alleges that this is disclosed by Mariani. A reference which does not even contain the term "version control" certainly does not disclose a method or system wherein "wherein the unique identifier and a version identifier of objects in a source project that are similar to the unique identifier and a version identifier of objects in the

destination project." The Final Office Action relies on a "name in the scope of a class" to allegedly disclose a unique identifier. (Final Office Action, page 11). However, even if a name in a scope of a class disclosed a unique identifier (which it does not), a class is not a project. Furthermore, Mariani does not disclose copying objects between projects as is inherent in claim 7 by virtue of its dependency on claim 6.

For at least these reasons, the rejection of claim 7 should be overturned.

H. Claim 15 is Separately Patentable

Claim 15 recites an "operational module [which] communicates with the user interface to select whether to copy the selected object from the source project to the destination project, to replace an object in the destination object with the selected object, and to keep an object in the destination project as is." As discussed above, Mariani is not configured to copy objects in the manner by which it recompiles object code. Copying and recompiling are completely different operations that require completely different systems and methods. Thus even if Mariani taught the limitations of claim 10 (which it does not), the Mariani system cannot be combined with Almond to copy objects in the way that it selectively recompiles object code. Thus, the combination of Mariani and Almond does not teach or suggest claim 15.

For at least these reasons, the rejection of claim 15 should be overturned.

VIII. Conclusion

Because the cited references, taken either singly or in combination, fail to disclose the combinations set forth in the pending claims, and further fail to provide any motivation or suggestion of the desirability of modifying the structures or methods to arrive at the claimed combinations, appellant submits that the pending claims are allowable over the cited references. Accordingly, Appellant respectfully requests that the Board reverse the prior art rejections set forth in the Action.

Respectfully submitted,

February 6, 2007

Brian M. Buroker
Registration No. 39,125

Hunton & Williams
1900 K. St., NW, Suite 1200
Washington, D.C. 20006-1109
(202) 955-1894

IX. Claims Appendix

1. A computer implemented method for managing groups of objects for use in a reporting system project comprising the steps of:

receiving a command to perform a selected function on a selected object;

automatically identifying dependent objects referred to by the selected object;

determining using a computer processor an appropriate manner of executing the selected function on the selected object;

determining using a computer processor appropriate functions to be performed on the dependent objects;

automatically causing the appropriate functions to be performed on the dependent objects; and

automatically causing the execution of the selected function on the selected object in the appropriate manner.

2. The method of claim 1 wherein the selected object is contained in metadata of an on-line analytical processing system.

3. The method of claim 1 wherein the step of causing the appropriate functions to be performed on the dependent objects is performed prior to the step of causing the execution of the selected function in the appropriate manner.

4. The method of claim 1 wherein objects are grouped in projects and the selected function relates to manipulating objects within and between projects and wherein within each project each object has a unique identifier and a version identifier.

5. The method of claim 4 wherein the step of determining appropriate functions to be performed on the dependent objects includes the steps of:

comparing dependent objects at a source and objects at a destination to determine whether an object at the destination exists in an identical form to each of the dependent objects at the source and whether an object at the destination exists in a modified form to each of the dependent objects at the source; and

determining, based on the step of comparing, which dependent object to copy from the source to the destination such that the selected object remains complete after execution of the selected function in the appropriate manner.

6. The method of claim 4 wherein the step of receiving is a step of receiving a command to copy a selected object from a source project to a destination project.

7. The method of claim 6 wherein the unique identifier and version identifier of objects in the source project are similar to the unique identifier and version identifier of objects in the destination project.

8. The method of claim 6 wherein the step of determining an appropriate manner of executing the function includes the steps of:

determining, by comparing unique identifiers and version identifiers, whether the selected object exists in the destination project in an identical form and whether the selected object exists in the destination project in a modified form; and

selecting whether to copy the selected object from the source project to the destination project, to replace an object in the destination project with the selected object, and to keep an object in the destination project as is.

9. The method of claim 8 wherein the step of selecting includes prompting the user for a selection.

10. A system application for managing objects within and between projects of a reporting system, the objects including unique identifiers and version identifiers that are similar between projects, the system application comprising:

a user interface for receiving a user command to manipulate a selected object; and

an operational module interfacing with the projects for identifying dependent objects referred to by the selected object, determining an appropriate manner of executing the user command, determining appropriate functions to be performed on the dependent objects, performing the appropriate functions on dependent objects, and executing the user command in the appropriate manner.

11. The system application of claim 10 wherein the operational module interfaces with projects that reside in various environments.

12. The system application of claim 10 wherein the operational module performs the appropriate functions on dependent objects prior to executing the user command in the appropriate manner.

13. The system application of claim 10 wherein the operation module interfaces with projects of an on-line analytical processing system.

14. The system application of claim 10 wherein the operational module, upon receiving a user command to copy the selected object from a source project to a destination project, determines, by comparing the unique identifiers and the version identifiers, whether the selected object exists in the destination project in an identical form and whether the selected object exists in the destination project in a modified form.

15. The system application of claim 14 wherein the operational module communicates with the user interface to select whether to copy the selected object from the source project to the destination project, to replace an object in the destination object with the selected object, and to keep an object in the destination project as is.

16. The system application of claim 10 wherein the operational module compares dependent objects at a source project and objects at a destination project to determine whether an object at the destination exists in an identical form to each of the dependent objects at the source and whether an object at the destination project at the destination exists in a modified form to each of the dependent objects at the source.

17. The system application of claim 16 wherein the operational module selects dependent objects from the source project to copy to the destination project such that the selected object remains complete after execution of the user command in the appropriate manner.

18. A processor-readable medium including code for execution on a processor to managing groups of objects comprising:

code for receiving a command to perform a selected function on a selected object;

code for identifying dependent objects referred to by the selected object;

code for determining an appropriate manner of executing the selected function;

code for determining appropriate functions to be performed on the dependent objects;

code for causing the appropriate functions to be performed on the dependent objects; and

code for causing the execution of the selected function in the appropriate manner.

X. Evidence Appendix

None.

XI. Related Proceedings Appendix

None.